Journal of Computer Science 2 (10): 798-806, 2006
ISSN 1549-3636

Cluster Computing: A Mobile Code Approach

R.B.Patel and Manpreet Singh
Department of Computer Engineering, M.M. Engineering College, Mullana-133203, Haryana, India

Abstract: Cluster computing harnesses the combined computing power of multiple processors in a parallel configuration. Cluster Computing environments built from commodity hardware have provided a cost-effective solution for many scientific and high-performance applications. In this paper we have presented design and implementation of a cluster based framework using mobile code. The cluster implementation involves the designing of a server named MCLUSTER which manages the configuring, resetting of cluster. It allows a user to provide necessary information regarding the application to be executed via a graphical user interface (GUI). Framework handles- the generation of application mobile code and its distribution to appropriate client nodes, efficient handling of results so generated and communicated by a number of client nodes and recording of execution time of application. The client node receives and executes the mobile code that defines the distributed job submitted by MCLUSTER server and replies the results back. We have also the analyzed the performance of the developed system emphasizing the tradeoff between communication and computation overhead.

Key words: Middleware, mobile code, cluster computing

INTRODUCTION

Cluster computing is an important element in mainstream computing. In recent years, cluster computers have emerged as the leaders in high-performance computing. Cluster computing harnesses the combined computing power of multiple microprocessors in a parallel configuration. Cluster computers are a set of commodity PC's dedicated to a network designed to capture their cumulative processing power for running parallel-processing applications[1]. Clustered computers are specifically designed to take large programs and sets of data and subdivide them into component parts, thereby allowing the individual nodes of the cluster to process their own individual chunks of the program.

A Cluster is a group of loosely coupled computers that work together closely so that in many respects they can be viewed as though they are a single computer. Clusters are commonly, but not always, connected through fast local area networks. Clusters are usually deployed to improve speed and/or reliability over that provided by a single computer, while typically being much more cost-effective than single computers of comparable speed or reliability. Cluster can be categorized into three forms.

High-availability (HA) clusters are implemented primarily for the purpose of improving the availability of services which the cluster provides. They operate by having redundant nodes, which are then used to provide service when system components fail. The most common size for an HA cluster is two nodes, which is the minimum required to provide redundancy. HA cluster implementations attempt to manage the redundancy inherent in a cluster to eliminate single points of failure.

Load balancing clusters operate by having all workload come through one or more load-balancing front ends, which then distribute it to a collection of back end servers. Although they are implemented primarily for improved performance, they commonly include high-availability features as well. Such a cluster of computers is sometimes referred to as a server farm.

High-performance clusters (HPC) are implemented primarily to provide increased performance by splitting a computational task across many different nodes in the cluster and are most commonly used in scientific computing. One of the most popular HPC implementations is a cluster with nodes running Linux as the system and free software to implement the parallelism. This configuration is often referred to as a Beowulf cluster[2]. Such clusters commonly run custom programs which have been designed to exploit the parallelism available on HPC clusters. Many such programs use libraries such as MPI[3] which are specially designed for writing scientific applications for HPC computers. HPC clusters are optimized for workloads which require jobs or processes happening on the separate cluster computer nodes to communicate actively during the computation. These include computations where intermediate results from one node's calculations will affect future calculations on other nodes.

Corresponding Author: R.B.Patel, Department of Computer Engineering, M. M. Engineering College, Mullana-133203, Haryana, India

In this paper we have presented design and implementation of a cluster based framework using mobile code. The cluster implementation involves the designing of a server named MCLUSTER which manages the configuring, resetting of cluster. We have also the analyzed the performance of the developed system emphasizing the tradeoff between communication and computation overhead.

Basic architecture of cluster computing: There are many cluster configurations, but a simple architecture is shown in Fig.1.

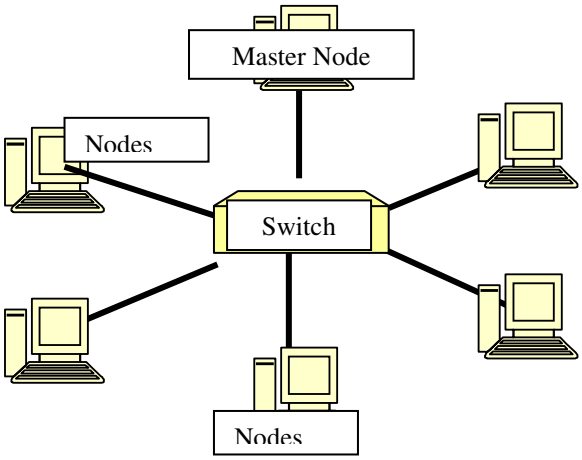

Fig. 1: Basic architecture of cluster computing

In a typical cluster, the application runs on a Master node. However, the computational work is split-up and parsed out to be done by the multiple nodes in the cluster. In this way, cluster is better equipped to handle larger amounts of data and complex problems than otherwise possible on a stand-alone machine. Some of the main modules related to cluster configuration are: Building a Cluster, System Administration, Hardware Management and Software Platform Maintenance.

After the cluster is constructed, it requires an effective system administration to remain useful. Maintenance and administration of a cluster are similar to those of a LAN. Two major domains of work explored in this area are: hardware management and software platform maintenance.

The main component of Hardware management is network management which can be divided into two major areas: cabling and topology. Every machine in a cluster must be able to work with the other machines. Maintaining the software on a cluster consists of administrative work multiplied by 'n' nodes - each of which is potentially dependent on other nodes.

Parallel processing using cluster computing: Parallel processing mainly involves concurrent use of multiple processors to process data. Significant development in Network technology is paving a way for parallel processing. Cluster computing implements MIMD (Multiple Instruction Multiple Data Stream) model of Flynn's classification[4] of computer architecture using general purpose processors or multicomputers. Clusters are also suitable than special parallel computers for the execution of parallel applications as they can easily integrate into existing networks. By sharing the computers of owner-users, which are normally not accessible in a non-dedicated cluster, parallel applications can gain extra processing power to perform CPU-hungry computations. On the other hand, owner-users of their computers could suffer from a slight degradation of the execution performance. The degradation of the CPU services trends to be insignificant when the workload of the computers move towards I/O-bound applications and the number of owner-users is large in the cluster.

Design of cluster computing: The main aim is to provide a flexible framework for Cluster Computing. The framework so used consists of three parts: Personal Computers (PC), high speed communication network, distributed applications as shown in Fig. 2.

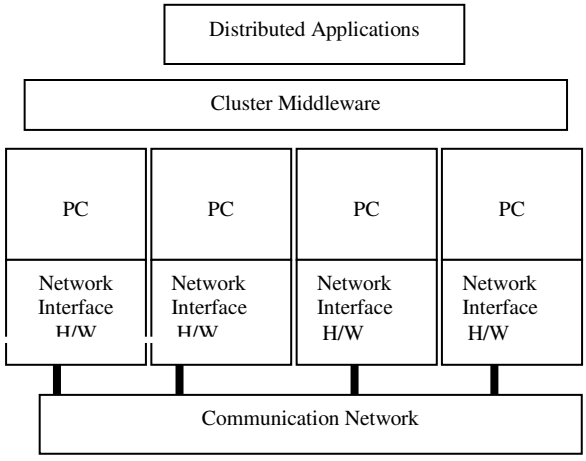

Fig. 2: Framework for Cluster Computing

PC are connected to the network using standard Ethernet Network Interface Card (NIC).Cluster middleware is implemented in Java so that middleware can provide the Single System Image[5] of the cluster to any computer with different OS platforms once the Java virtual machine(JVM) is installed. JVM makes it easier to implement, migrate and execute the mobile code at remote computer in the cluster. The user is guided through the creation and management of cluster via a graphical user interface. It frees the user from identifying the network topology of the framework of cluster. The framework has been designed in such a way that incremental changes to it can easily enhance the generality and usability of cluster.

Architecture of cluster middleware: In current cluster configuration there are two types of nodes:

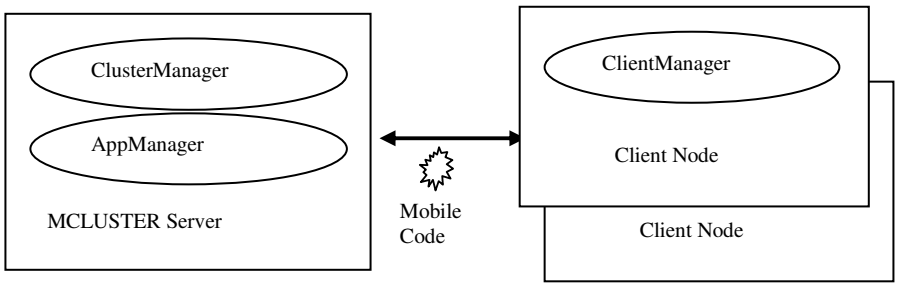

Fig. 3: Architecture of cluster middleware

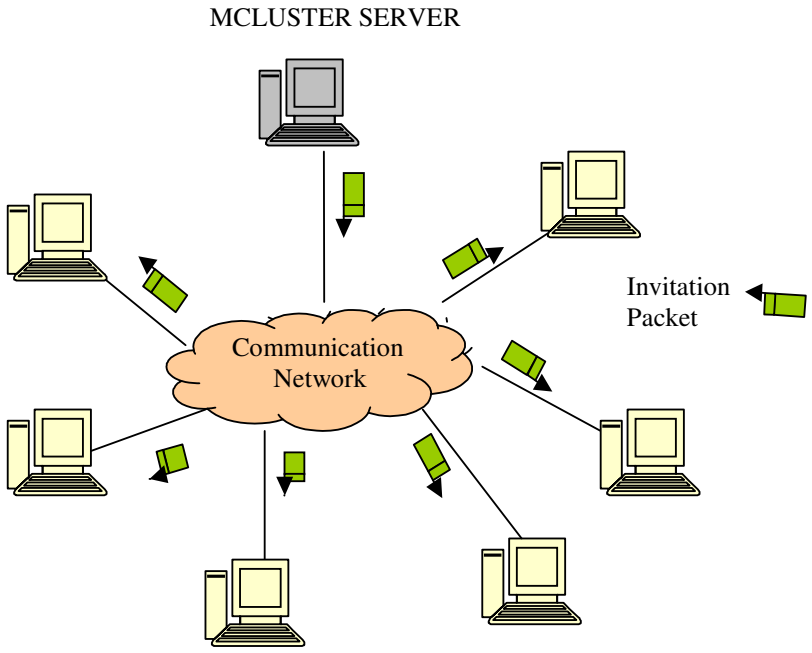

Fig. 4: Broadcast of Invitation Packet by MCLUSTER

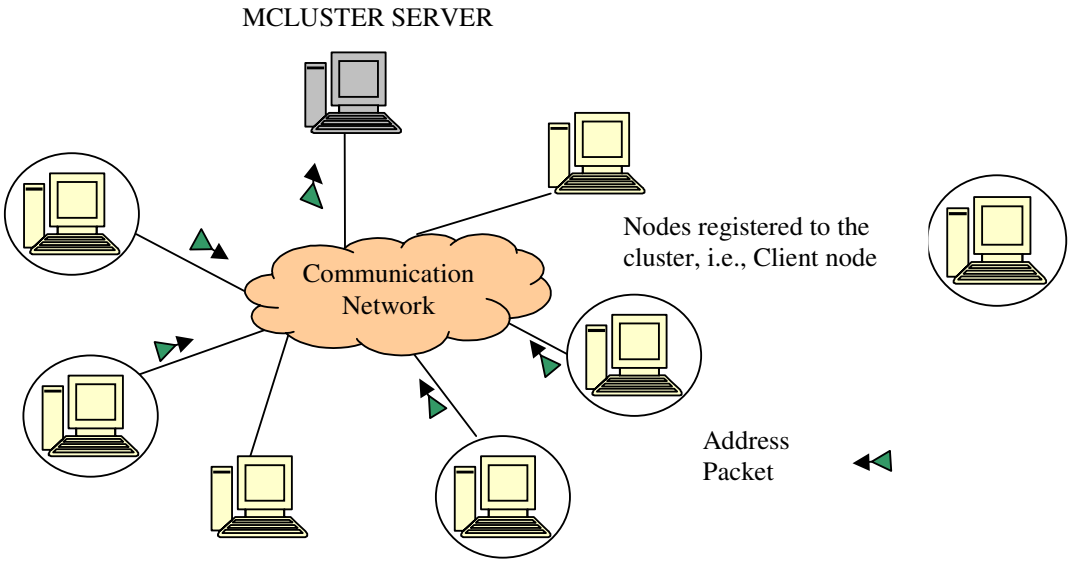

Fig. 5: Transmission of IP address from Client to MCLUSTER

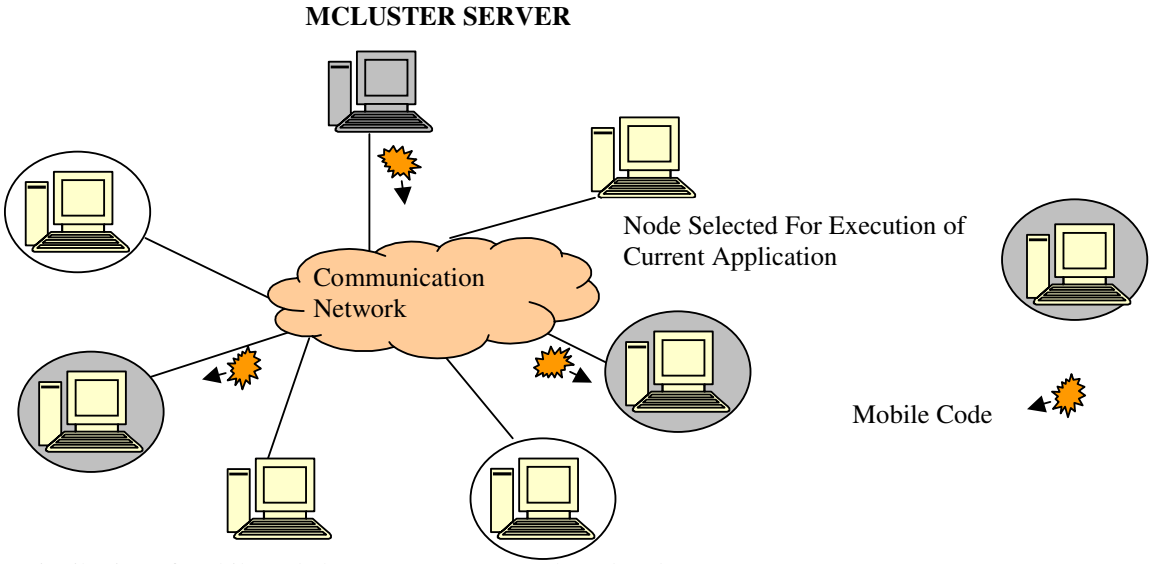

Fig. 6: Distribution of mobile code by MCLUSTER to selected nodes

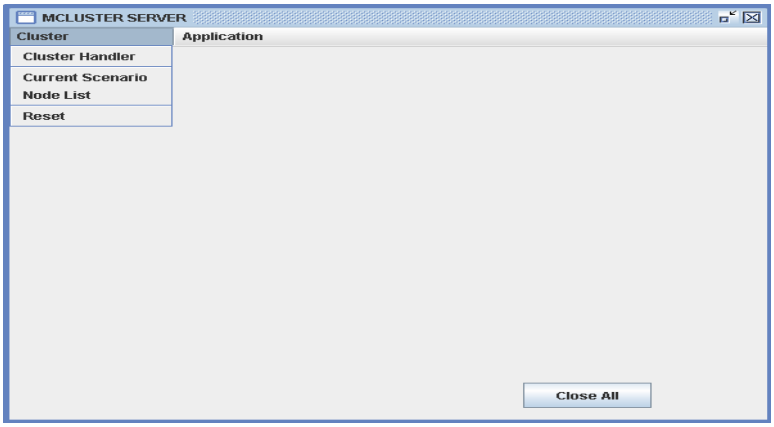

Fig .7: Options related to cluster management

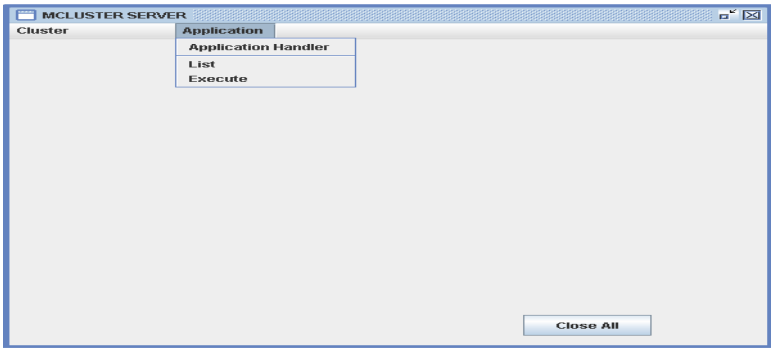

Fig. 8: Options related to application description and execution

a server node named MCLUSTER, client nodes. Middleware is generally considered the layer of software sandwiched between the operating system and applications. Cluster middleware allows programmer to develop and execute the parallel applications on clusters and achieve good execution performance. Figure 3 illustrates the structure of cluster middleware.

MCLUSTER server and client nodes are communicating with each other through message passing and any distributed application is executed using mobile code which contains both input data as well as application code. The important building blocks of cluster middleware are ClusterManager, AppManager and ClientManager.

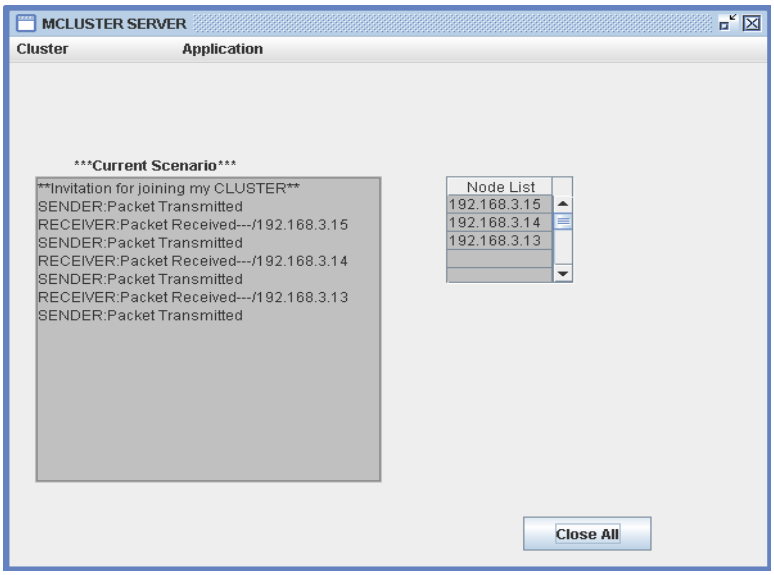

Fig. 9: Selection of current scenario and node list option

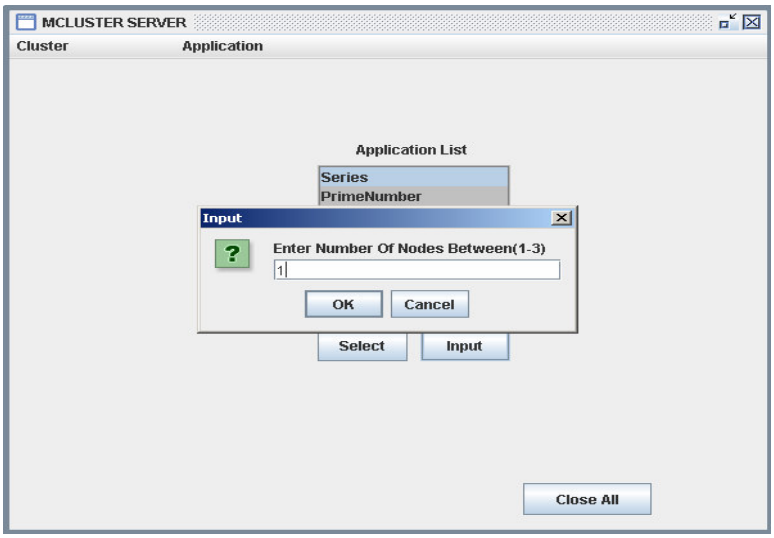

Fig. 10: Selection of input button for specifying number of nodes to be selected

MClusterServer

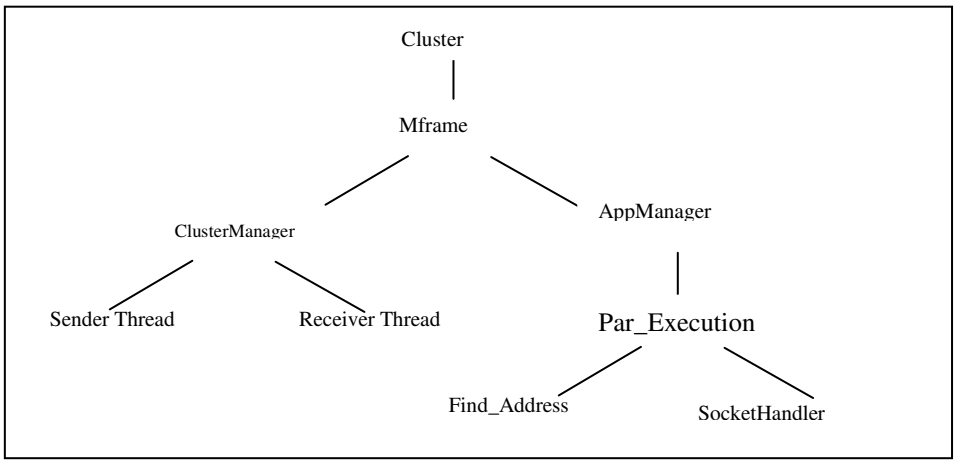

Fig. 11: Class diagram of middleware implementation of MCLUSTER SERVER

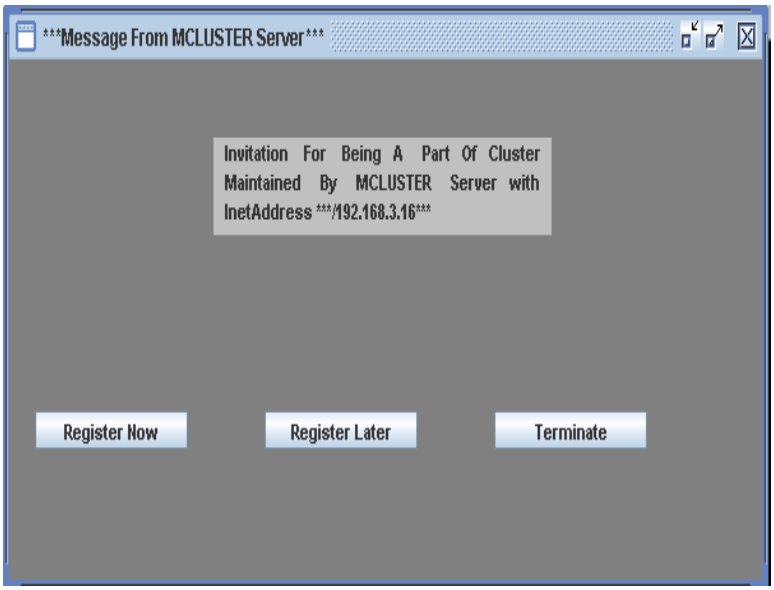

Fig. 12: Frame displayed at client node corresponding to invitation packet

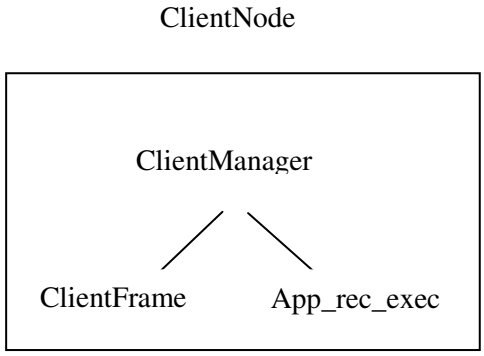

Fig. 13: Class diagram of middleware implementation of client node

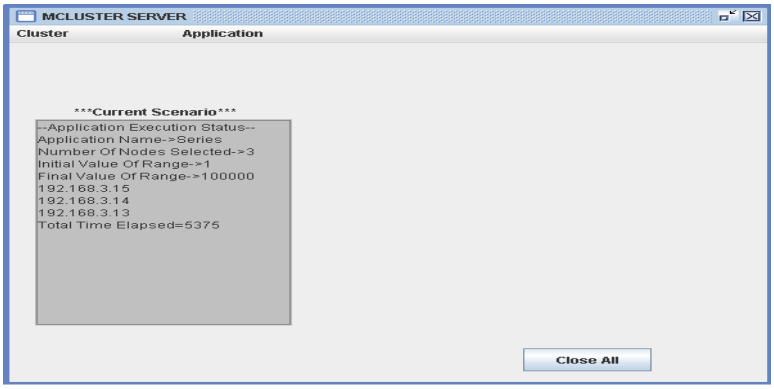

Fig. 14: Result of application series with client nodes=3 and data range=1-100000

ClusterManager is the software module that initiates the cluster and manages all nodes that take part in the cluster process. ClusterManager periodically broadcast invitation packets to all nodes in the network as shown in (Fig. 4). All those nodes on which ClientManager is activated and are not being part of the cluster, will display a frame on the user screen after receiving the invitation packet .If user accepts the invitation then IP address of user's node will be automatically communicated to MCLUSTER Server as show in Fig. 5. MCLUSTER efficiently maintains a database of all these addresses.

AppManager is the module which coordinates all functions related to application description, distribution and execution. AppManager provides a GUI from which user can easily select a particular application,

provides the input data and specify the parameters such as number of nodes to be used for the execution of application. After the selection of application, AppManager initiates Par_Execution module. Par_Execution module implements an algorithm for parallel execution of application. Steps of algorithm are as follows:

Select the first N IP Addresses from the database, where N is the number of nodes to be used for the execution of current application as specified by the user.

Divide the given data range into N number of non Overlapping data sub range.

Send an Execute Packet to all selected Client nodes.

Creates N number of threads where each thread is involved in the transfer of Mobile code as shown in Fig. 6 (class file of application, data sub range) to one of the selected client node as well as the retrieval of result from the same node.

ClientManager is the software that listens to the requests from MCLUSTER server related to cluster membership, application execution and service those requests ClientManager is the code which can be executed on any node in the network. During its execution, it will receive three types of messages from MCLUSTER server: Invitation message, Execution message and Reset message. A node can provide its Computational power to cluster only when it register itself to MCLUSTER by communicating its IP address to the server in lieu of Invitation message & then it becomes one of the client node of cluster. On receiving the mobile code from server it load and execute the mobile code and reply the results back.

Implementation and performance study: There exist two main classes that constitute the cluster middleware. Cluster class creates an object of Mframe class which in turn builds a GUI that helps the user to obtain cluster as well as application execution information interactively as shown in Fig. 7 and 8.

Mframe also coordinates with ClusterManager and AppManager classes. ClusterManager class contains two supporting classes: Sender and Receiver mainly involved in cluster monitoring[6] such as building and resetting as shown in Fig. 9. AppManager class receives the application name, data range; node count information from the user via GUI as shown in Fig. 10. This information is used by its two subclasses: SocketHandler and Find_Address for the generation and distribution of mobile code to the required client nodes.

ClientManager mainly provides communication with MCLUSTER server. It contains two supporting classes: ClientFrame and App_rec_Exec. ClientFame is providing a GUI to the user as shown in Fig. 12 while App_rec_Exec provides an environment for the execution of mobile code.

Series application is selected for validating the prototype. It basically involves generating a sequence of numbers between an initial value and a final value. The job load is evenly distributed among selected client nodes. MCLUSTER server begins a timer before the execution of application and stops it after the collection of entire series from client nodes as shown in Fig. 14. The effect of number of client nodes and data range on the execution time of application is represented by Fig. 15 and 16.

The performance of cluster, during the execution of distributed applications not having significant amount of data (such as between 1-20000) deteriorate as indicated by Overhead point in Fig. 15. At these data ranges the communication overhead between MCLUSTER Server and Client nodes overwhelm the advantage of distributed processing power obtained from client nodes. When data range is extensive then the time consumed by an application so executed in a distributed manner is several times smaller than the execution time of the same application processed on a single node. The MCLUSTER model is designed in such a way that even for a large problem size (such as between 1-5000000) and average cluster size (up to N=11) performance of distributed application will not deteriorate as shown in Fig. 16.

Overhead analysis: When the scalability of cluster increases i.e. more and more nodes are introduced in the cluster, the communication links near the server are congested due to large transmission of data thereby degrading the cluster performance. Communication overhead includes the overhead due to exchange of data between nodes and message delay caused by network congestion. The result so generated after execution of the application Series for a data range 1-100000 on a cluster of 11 client nodes is represented by Fig. 17.

From Fig. 17 and on the basis of assumption that external network load is low and constant ,the communication overhead for MCLUSTER model can be parameterized as a simple linear function of number of bytes transmitted and number of client nodes selected.

$C_{overhead} = \alpha + \beta * N$

Where α=Startup time involving Partitioning time, Time spent during selection
of client nodes.
β= Cost per byte transmitted.
N=Number of nodes selected for a particular execution

Related work: A number of research efforts in the area of improving the performance of distributed applications in a cluster computing environment have emerged[2,7,8].Previous studies revealed the impact of cluster size and underlying network on the performance of distributed applications[9,10].

Our work (i) evaluates the performance of application as a function of problem size, network

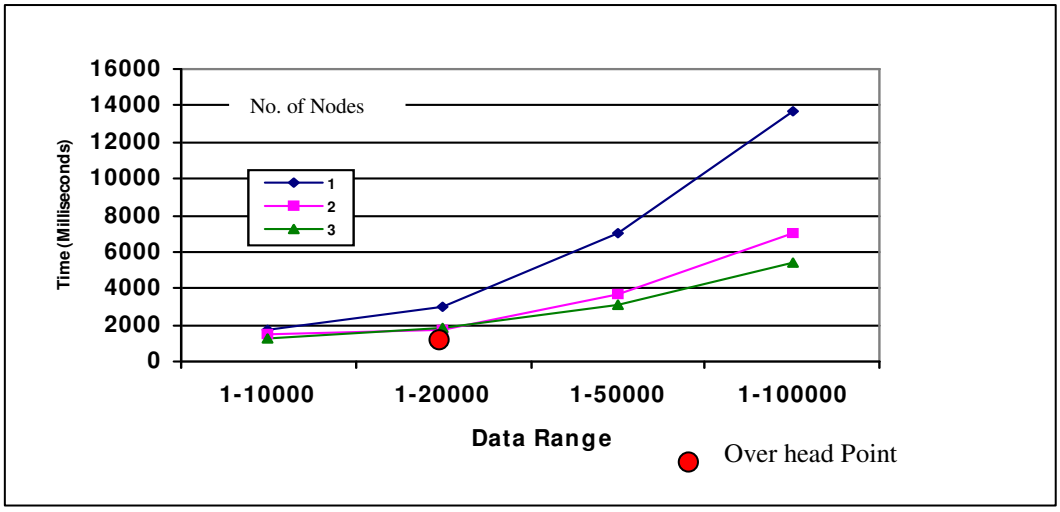

Fig. 15: Effect of data range and number of nodes on execution time

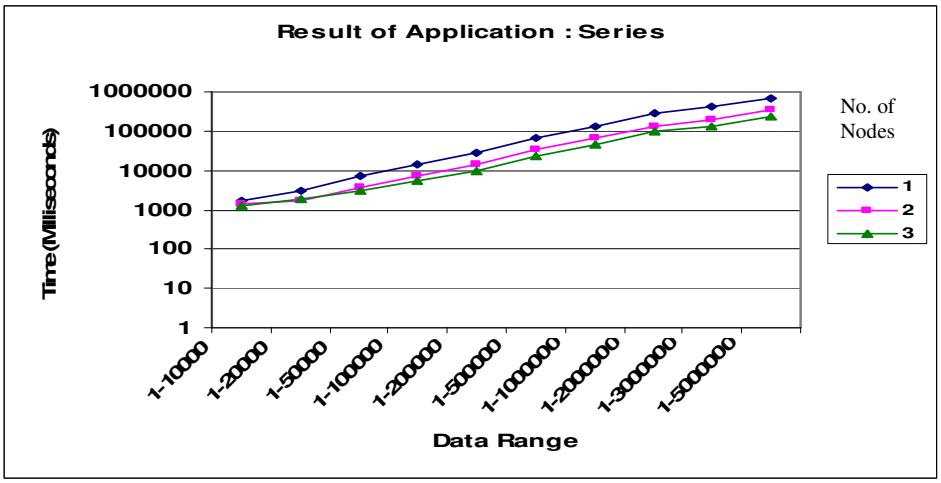

Fig. 16: Effect of data range on execution time of application series

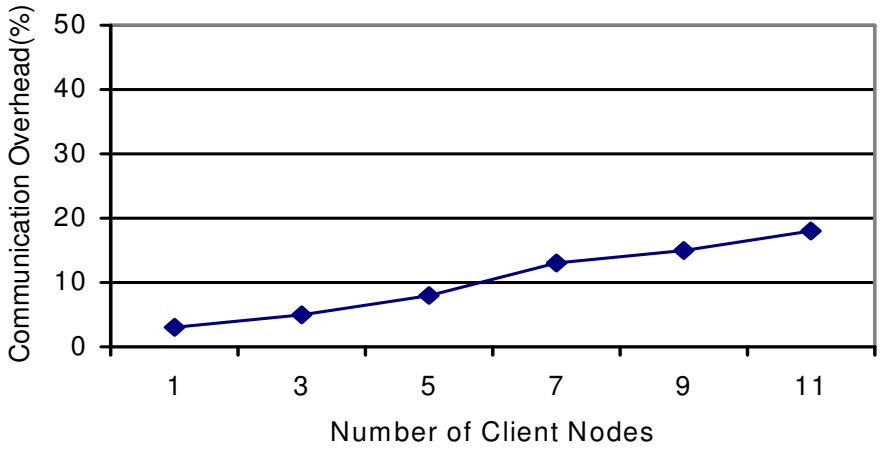

Fig. 17: Communication overhead vs. cluster size

parameters and cluster size (ii) provides features for monitoring of cluster. The developed MCLUSTER model concentrates on reducing the communication overhead by using the mobility of code and bridging all system characteristics.

CONCLUSION AND FUTURE WORK

Cluster computing undoubtedly is gaining importance as a substitute for very expensive Parallel computers. As depicted in my dissertation work, Cluster computing by appropriately combining the computational processing power of autonomous computers can significantly improve the performance of distributed applications. Cluster Computing on the other hand can deteriorate the performance of a problem if a proper check is not applied at the problem size as well as cluster size. Since the improper values of these two parameters will lead to the network congestion thereby resulting in the overwhelming of computation load by communication load. However this Network Congestion problem can be easily tackled by using the concept of Mobile Agent[11].

Further we will implement Fault Tolerance in MCLUSTER model, i.e., to make sure that cluster functionality is not largely affected even if MCLUSTER server fails. It can be attained by replicating the functionalities of MCLUSTER server on some other nodes in the cluster.

REFERENCES

1. Baker, M. and R. Buya, 1999. Cluster computing: The commodity supercomputer. Software-Prentice, 29: 551-576.
2. Becker, D. and P. Merkey. The Beowulf Project. http://www.beowulf.org.
3. Message Passing Interface, MPI Forum-http://www.mpi-forum.org.
4. Flynn, M.J. Very High Speed Computing Systems. Proc. IEEE, 54: 1901.
5. Geist,A. and J. Schwidder, 1999. Managing multiple multi-user PC clusters. J. Parallel and Distributed Computing.
6. Cheung, L. and A.P. Reeves, 1992. High performance computing on a cluster of workstations. Proc. First Symp. High-Performance Distributed Computing.
7. Jon, B.W. and A.S. Grimshaw, 1994. Network partitioning of data parallel computations. Proc. Third Intl. Symp. High Performance Distributed Computing (HPDC '94), April 2-5, 1994, San Francisco, CA, USA. IEEE Computer Society.
8. Lemeire, J. and E. Dirkx, 2002. Causes of blocking overhead in message-passing programs. 10th Euro PVM/MPI 2002 Conference, Venice, Italy.
9. Martin, R., A. Vahdat, D. Culler and T. Anderson, 1997. Effects of communication latency, overhead and bandwidth in a cluster architecture. Proc. 24th Annual Intl. Symp. Computer Architecture (ISCA), pp: 85-97.
10. Lange, D.B. Mobile Agents: The Future of Distributed Computing? General Magic, Inc. California.
11. Walker, B. and D. Steel, 1999. Implementing a full single system image unixware cluster: Middleware vs. underware. Proc. Intl. Conf. Parallel and Distributed Processing Techniques and Applications(PDPTA'99),Las Vegas,USA.